HeidelPlace: An Extensible Framework for Geoparsing

Ludwig Richter and **Johanna Geiß** and **Andreas Spitz** and **Michael Gertz**
Institute of Computer Science, Heidelberg University
Im Neuenheimer Feld 205, 69120 Heidelberg
ludwig.richter@posteo.de
{geiss,spitz,gertz}@informatik.uni-heidelberg.de

Abstract

Geographic information extraction from textual data sources, called *geoparsing*, is a key task in text processing and central to subsequent spatial analysis approaches. Several geoparsers are available that support this task, each with its own (often limited or specialized) gazetteer and its own approaches to toponym detection and resolution. In this demonstration paper, we present HeidelPlace, an extensible framework in support of geoparsing. Key features of HeidelPlace include a generic gazetteer model that supports the integration of place information from different knowledge bases, and a pipeline approach that enables an effective combination of diverse modules tailored to specific geoparsing tasks. This makes HeidelPlace a valuable tool for testing and evaluating different gazetteer sources and geoparsing methods. In the demonstration, we show how to set up a geoparsing workflow with HeidelPlace and how it can be used to compare and consolidate the output of different geoparsing approaches.

1 Introduction

The ever-growing amount of available text data raises the need for automated Information Extraction (IE) to obtain structured information from text. A central step in such an extraction procedure is the semantic annotation of the contained information (Zhang and Rettinger, 2014). For tasks such as event detection or content analysis, temporal and spatial information play a crucial role. In particular, the linking of place mentions to geographic databases, so-called *geoparsing*, is an integral element of textual analyses (Andogah et al., 2012). Geoparsing describes the process of identifying place mentions in text (so called *toponyms*) and linking them to unambiguous spatial references. Consider the example text *"Heidelberg was founded in 1196 AD"*, which contains the place mention *"Heidelberg"*. Typically, geoparsing involves three components: *gazetteers*, *toponym recognition* and *toponym resolution*. A Gazetteer is a dictionary of geographic features that serves as a knowledge repository for places of interest. Toponym recognition deals with the detection of place names in documents. For instance, *"Heidelberg"* can be detected by matching candidate strings to a gazetteer of city names. With the help of toponym resolution (also called *toponym linking*), each identified toponym is then matched to an unambiguous spatial reference (e.g., latitude and longitude coordinates). In our example above, the spatial reference may be taken from the matched gazetteer entry. Due to ambiguous names of mentioned places, this task often requires *toponym disambiguation*. For instance, *"Heidelberg"* refers to several places world-wide. If the gazetteer contains founding dates, we can infer that the text refers to Heidelberg in Germany.

Linguistic peculiarities, as well as the relevance of different places and place features, greatly vary with the application domain. Hence, geoparsing is a non-trivial process that requires fine tuning according to the respective use-case. As a result, a variety of geoparsing approaches have been proposed (Hoffart, 2015; Leidner, 2007; Lieberman and Samet, 2011, 2012), many of which are domain-specific. There exist several specialized gazetteers, such as GeoNames[1], a large gazetteer for a broad spectrum of place types, OpenStreetMap[2], a community-built geo-

[1] http://geonames.org
[2] http://openstreetmap.org

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP '17), September 7-11, 2017, Copenhagen, Denmark

graphic knowledge-base, and Pleiades[3], a historic gazetteer. However, reusing and comparing existing geoparsers is often difficult, since they generally come with their own gazetteer and processing pipeline. Configurability and extensibility are rarely considered. As a result, it is often problematic to adjust a geoparser to other application domains or to include other algorithms. Especially the adjustment to a different gazetteer source tends to be troublesome. Each gazetteer uses its own data model and most geoparsers use the data model of their primary gazetteer resource. If the underlying gazetteer data model is not designed to be flexible, or if the geoparser is too tightly coupled with a specific gazetteer source, a reuse may not be possible.

CLAVIN[4], an open-source geoparsing framework, strives to overcome these issues by providing an extensible toolbox of geoparsing methods. However, the framework strongly relies on GeoNames as gazetteer, which is often not an ideal choice. Switching to another data source would entail a significant rewrite of the code base.

Thus, a generalized gazetteer data model is required that is compatible with different gazetteer sources. Only then can significant software modifications due to gazetteer specific data modeling be avoided, and the adaptation work be reduced to the transformation of data sources to the generic data model. While CLAVIN's modular design allows to add new modules, its processing pipeline is too restrictive to enable complex geoparsing approaches that rely on information exchange between different modules.

Contributions. In this work, we present HeidelPlace, a geoparsing framework that includes a generic gazetteer model and an implementation of the entire geoparsing process. The generic gazetteer model supports the integration and management of heterogeneous, large-scale gazetteer sources, and is flexible enough to integrate concepts from different gazetteers. A variety of concepts proposed in gazetteer research are included (Hill, 2000; Keler et al., 2009; Moura and Davis, 2014), which allow the description of places in a comprehensive and yet flexible manner.

The extensibility of HeidelPlace is realized in two ways. First, modularization is a central design aspect, with different geoparsing approaches being represented as modules. They can be used and combined in a plug-in like manner, allowing the integration of newly developed modules. Second, interactions between the different components are handled transparently by the framework, using an annotation-based processing pipeline. Therefore, the user can focus on obtaining appropriate configurations through experimentation. To further increase the efficiency of developing new geoparsing methods, we include user-friendly GUIs that make the geoparsing process transparent. The open-source project HeidelPlace and the data used for this demonstration are available for download from the EventAE project website[5].

2 Framework and System Overview

HeidelPlace includes a geoparsing framework for developing new geoparsers as well as GUIs to make the geoparsing process transparent to the user. The architecture of our proposed geoparsing framework is visualized in Figure 1, which highlights the three major components that we describe in the following. The GUIs are introduced towards the end of this section and in Section 3.

Annotation Pipeline: In order to execute the geoparsing process (either entirely or partially) on input documents, we utilize the annotation pipeline of the Stanford CoreNLP toolkit (Manning et al., 2014), which is built atop three key interdependent concepts, namely `Annotation objects`, `Annotators`, and `AnnotationPipelines`. An `Annotation object` stores key-value information about a document with arbitrary structure. As depicted in Figure 1, this may be the original text of a processed document, identified tokens, or mentions of named entities. An `Annotator` is a configurable module that performs some processing task on a given document. It can read and write analysis information from and to the `Annotation objects`, allowing to pass information among different `Annotators`. The annotator does not implement the processing task itself, but delegates between different modules that implement the same functionality. For instance, `Tokenizer` modules scan the original text of a document with specific strategies and return a set of identified tokens. A `TokenizerAnnotator` is configured by the user to utilize a particular `Tokenizer`

[3] http://pleiades.stoa.org
[4] https://clavin.bericotechnologies.com
[5] http://event.ifi.uni-heidelberg.de/?page_id=517

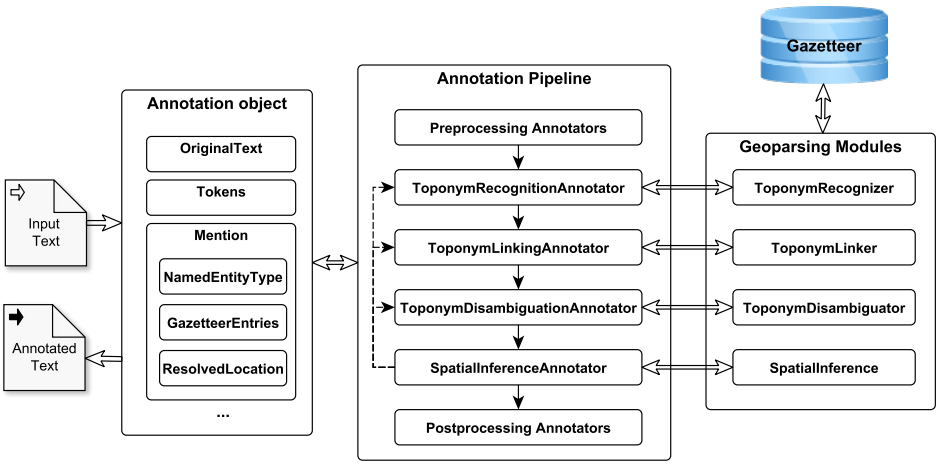

Figure 1: The geoparsing framework architecture, including the three major components: annotation pipeline, geoparsing modules, and gazetteer.

module. It passes all required information to the module and adds annotations for retrieved output tokens. This additional layer of abstraction allows to separate pipeline maintenance tasks from the implementation of the actual data processing methods. The `AnnotationPipeline` consists of a set of `Annotators`, which are run sequentially over the input document.

Geoparsing Modules: The main objective of our framework is the unification of the geoparsing process, with a focus on extensibility and usability. Therefore, we use a modular design and an easy-to-use processing pipeline as described above. The geoparsing process is divided into four steps, which are represented by modules:

- `ToponymRecognizer`: Extracts toponyms from a given input text document.
- `ToponymLinker`: Links each toponym to places in the gazetteer. Due to ambiguity, multiple matches per toponym are possible.
- `ToponymDisambiguator`: Resolves ambiguous matches.
- `SpatialInference`: Infers the location of unlinked toponyms.

For each step, an `Annotator` feeds the correct input to the respective module and then adds its output to the `Annotation object`.

The processing steps need to be executed in sequence, but may be re-run multiple times. For example, running a disambiguation process before the entity recognition step does not make sense. However, re-running a recognition module after a first complete geoparsing pass may improve the results in the second run.

Gazetteer: We developed a generic gazetteer model that allows to incorporate a wide spectrum of place related information. Its central concept is a *geographic place*, which may have multiple names, footprints, types, properties, and relationships, depending on the available data. This allows us to incorporate data from multiple heterogeneous sources.

To enable multilingual and context sensitive geoparsing, we allow each place name to have a language entry and a set of flags (e.g., *is historic* or *is official*). By supporting multiple footprints per place, different spatial representations like points, lines, or polygons are possible. This enables the use of different geographic resolutions and allows us to capture irregularities such as historic changes or political border conflicts. Since a place may fulfil different functions, multiple types can be assigned to each geographic place. In addition, each place may have an arbitrary number of properties with an assigned type and value (e.g., its *population* or a *Wikipedia link*). To link places within the gazetteer, typed place relationships can be defined (e.g., *sister city* or *topographic neighbor*) and used to implement administrative hierarchies. Optionally, relationships may have a value assigned, e.g., a place co-occurrence score (Overell and Rüger, 2008; Spitz et al., 2016). With a flexible type scheme, complex ontologies for place types, property types, and relationship types can be created by using *parent-child* and *similar-to* relationships among types. This allows the integration of type systems of existing gazetteers, while maintaining expressiveness by linking related types. Figure 2

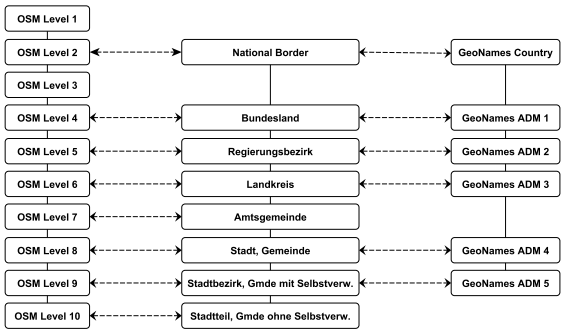

Figure 2: Mapping of GeoNames and OSM administrative levels to a German administrative hierarchy in the HeidelPlace gazetteer model.

shows an exemplary mapping of GeoNames and OSM administrative levels to administrative hierarchies in the `PlaceType` entity class of the HeidelPlace gazetteer model. Furthermore, *provenance* and *validity times* can be recorded for each place and its descriptors. This allows to document data origin and temporal validity such as the change in population, which is especially important for historic text corpora.

To support easy access to the gazetteer database, we provide an API written in Java using JPA. This includes a mapping of the database schema to Java classes as well as a flexible query interface. It serves as an abstraction layer for complex database-level access to the underlying data model. A set of predefined filters for each place descriptor can be used to narrow the place search. Users may implement more filters through the JPA Criteria API. For advanced use cases, plain SQL queries support low-level access if desired.

Gazetteer Creation and Integration: Before we can perform actual geoparsing, we first need to fill our gazetteer with data. To demonstrate the applicability of the proposed gazetteer model, we implemented the model in a relational database schema using PostgreSQL 9.4 with PostGIS 2.2. We developed an importer for GeoNames to illustrate how the transformation and integration of data into our model works. A subset of the GeoNames dump from September 7th, 2016 was imported into the gazetteer database. It consists of 4.6 million places, with high coverage of higher-level administrative divisions and many populated places. We obtain basic POI coverage by adding hotels, airports, and castles. More than 6.3 million place names were extracted, 3.8 million of which are distinct (1.66 places per toponym, 1.38 toponyms per place). We use this data for the demonstration scenario. Due to the complexity of merging gazetteers, we refrain from elaborating on the conflation of gazetteers in this demonstration.

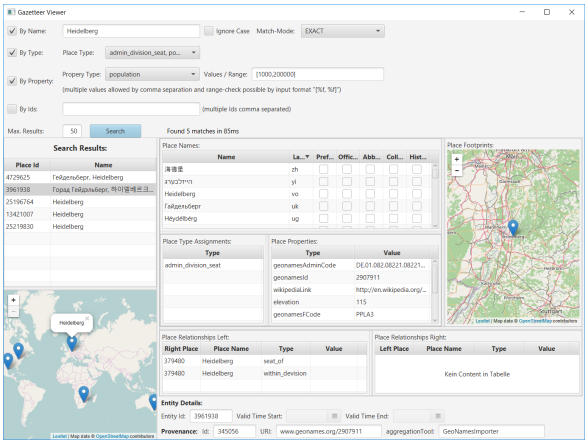

Figure 3: The gazetteer browsing GUI. The search filters can be specified at the top. On the left side the places are listed and shown on a map. The data for a selected place is displayed on the right side.

3 Demonstration Scenario

In the following, we describe the demonstration of four key usage scenarios of the framework.

Gazetteer Browsing: To make the geoparsing process transparent to the user, visual feedback is important. For the gazetteer component, we provide a user-friendly GUI called “Gazetteer Viewer”, which was developed using JavaFX (see Figure 3). The GUI allows users to conveniently browse the gazetteer data and aids them in understanding the underlying data model. The user can narrow search results by specifying a set of filters in the filter mask in the top area. By clicking ”Search”, the gazetteer is scanned for places that match the specified criteria. A list of identified places and an overview map with their locations are displayed on the left side. If a place is selected in the list or map, its details are shown on the right side. To follow links between places, double clicking on an entry in the relationship table opens a separate view with details of the related place.

With this tool, data exploration can be greatly improved. Neither manual database queries nor knowledge about the database layout are required. Furthermore, the data is clearly structured and visualized to quickly convey relevant information. To display the geoparsing process, we developed a second GUI as described in the following.

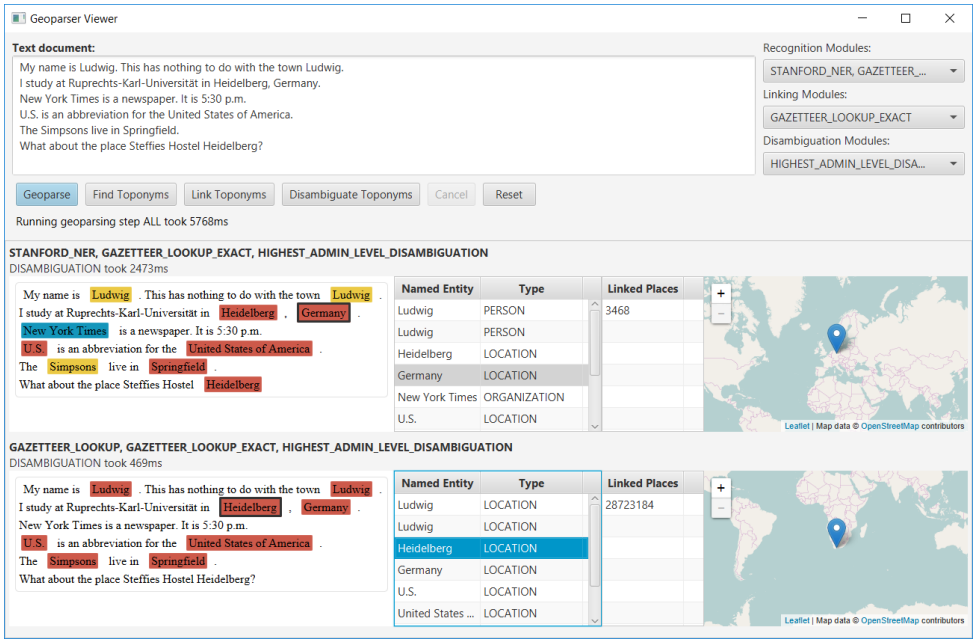

Figure 4: The geoparsing GUI. The input filed for text and the modules selection is located at the top. The annotations are shown below in an interactive area where the user can gain more information. For each geoparsing method a separate result view is shown.

Performing Geoparsing: The usage of the geoparsing framework is demonstrated with a set of exemplary geoparsing modules. Here, the focus lies on showcasing the interplay between the modules and not on methodical sophistication. For toponym recognition, we include three modules. The first module applies the location name finder of OpenNLP, while the second uses Stanford NER to detect location entities. The third module is based on gazetteer look-ups for proper nouns as well as basic linguistic considerations. A single toponym linking module is also provided, which links toponyms to gazetteer entries via an exact name matching filter. For toponym disambiguation, two basic modules are implemented. The first module gives precedence to places with higher population, whereas the second module gives precedence to pairs of candidate places that are geographically closer. Since spatial inference is a non-trivial task, we implement no exemplary inference module in this demonstration.

To exemplify how a geoparser can be implemented based on our geoparsing framework, we provide a JavaFX-based GUI called "Geoparser Viewer". The viewer is depicted in Figure 4. It supports a set of pre-configured modules and visualizes the geoparsing process. In the viewer, the user first provides an input text document and selects a module per geoparsing step. The geoparsing process can then be run by clicking "Geoparse". Alternatively, geoparsing can be executed step-by-step by clicking the respective buttons. Resulting annotations are visualized in the bottom part of the window (in Figure 4, results for multiple geoparsing methods are shown, as described in the last use-case). This visualization helps the user to analyze the geoparsing results, as we discuss in the next scenario. The simple replacement of modules highlights the benefit of a modular design and clearly structured processing pipeline.

Analyzing Geoparsing Results: If a geoparsing method produces unexpected output or if its behavior is not clear, detailed analysis of the process is required. The result view of the "Geoparser Viewer" allows to conveniently analyze the geoparsing results. On the left side, identified named entities are highlighted in the original text. Since geoparsers may use general NER tools like StanfordNER, our recognition module can also add annotations for named entities other than locations. Knowing the type of other entities may be of use for later processing steps (e.g., disambiguation). Therefore, each color represents a named entity type (among others, yellow for persons, red for locations, blue for organizations). Additionally, a list with all identified named entities is shown. Selections in the text are reflected in the list and vice versa. If a location entity is selected, a list of

linked places from the gazetteer is shown. A map displays the location for a selected linked place to quickly surmise the geographic context.

The results of the geoparsing process highly depend on the gazetteer data. For example, the type, administrative level, population, or mention frequency of a place may influence the recognition and disambiguation step. Hence, it may be of interest to see what information is stored for a linked place to draw conclusions about the geoparser decisions. By double-clicking on a linked place entry, a separate view is shown that is similar to the "Gazetteer Viewer". With such a visual analysis, experimentation and debugging of geoparsing methods is greatly simplified.

Comparing Geoparsing Methods: Another valuable feature is the comparison of different geoparsing modules. In the "Geoparser Viewer", multiple geoparsing methods can be compared qualitatively. If multiple modules per geoparsing step are selected, each combination from the cross product of all module combinations is considered a *geoparsing method*. After clicking "Geoparse" or any of the step-by-step buttons, a result view for each method is shown (see Figure 4).

The clearly structured and interactive visualization allows the user to quickly surmise the differences between several geoparsing methods. Of course, this does not replace quantitative performance analyses that allow for a more representative performance evaluation. Instead, it provides the means to easily test special cases and helps to better understand the geoparsing process and the employed modules.

4 Conclusion and Ongoing Work

In this paper, we presented HeidelPlace, a flexible and extensible geoparsing framework built on two paradigms. On the one hand, it introduces a generic gazetteer model that supports the integration of heterogeneous gazetteer sources. A comfortable query API and a user-friendly GUI aid users in maintaining and exploring the gazetteer. On the other hand, HeidelPlace utilizes a modularized pipeline for the entire geoparsing process. Since each geoparsing step is represented as a module within a flexible annotation-based processing pipeline, new approaches can easily be integrated and tested.

Our ongoing work includes the implementation of an evaluation component that enables a standardized quantitative comparison of different geoparsing approaches, the development of an UIMA component to simplify integration into existing IE pipelines, a web-service that supports remote queries to the gazetteer and geoparser, and the implementation of state-of-the-art modules as a starting point for a production-ready toolbox that can be improved and expanded by the community.

References

Geoffrey Andogah, Gosse Bouma, and John Nerbonne. 2012. Every Document Has a Geographical Scope. *Data Knowl. Eng.*, 81-82:1–20.

Linda L Hill. 2000. Core Elements of Digital Gazetteers: Placenames, Categories, and Footprints. *ECDL*, LNCS 1923:280–290.

Johannes Hoffart. 2015. *Discovering and Disambiguating Named Entities in Text*. Dissertation, Saarland University.

Carsten Keler, Krzyzstof Janowicz, and Mohamed Bishr. 2009. An Agenda for the Next Generation Gazetteer: Geographic Information Contribution and Retrieval. In *GIS '09*, pages 91–100.

Jochen L. Leidner. 2007. *Toponym Resolution in Text: Annotation, Evaluation and Applications of Spatial Grounding of Place Names*. Dissertation, University of Edinburgh.

Michael D. Lieberman and Hanan Samet. 2011. Multifaceted Toponym Recognition for Streaming News. In *SIGIR '11*, pages 843–852.

Michael D. Lieberman and Hanan Samet. 2012. Adaptive Context Features for Toponym Resolution in Streaming News. In *SIGIR '12*, pages 731–740.

Christopher D Manning, John Bauer, Jenny Finkel, Steven J Bethard, Mihai Surdeanu, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL '14*, pages 55–60.

Tiago H. V. M. Moura and Clodoveu A. Davis. 2014. Integration of Linked Data Sources for Gazetteer Expansion. In *GIR '14*, pages 5:1–5:8.

Simon E. Overell and Stefan M. Rüger. 2008. Using Co-Occurrence Models for Placename Disambiguation. *Int. Journal of Geographical Inf. Science*, 22(3):265–287.

Andreas Spitz, Johanna Geiß, and Michael Gertz. 2016. So Far Away and Yet So Close: Augmenting Toponym Disambiguation and Similarity with Text-based Networks. In *SIGMOD GeoRich '16*, pages 2:1–2:6.

Lei Zhang and Achim Rettinger. 2014. X-LiSA: Cross-lingual Semantic Annotation. *Proceedings of the VLDB Endowment*, 7(13):1693–1696.